REJ10J0927-0200(T)

Everywhere you imagine. RENESAS

# E8 Emulator

## Additional Document for User's Manual
## R0E000080KCE00EP4

Renesas Microcomputer Development Environment System
M16C Family / M16C/60 Series
Notes on Connecting the M16C/62P

User's Manual

Rev.2.00
Jun. 23, 2005

Renesas Technology
www.renesas.com

Keep safety first in your circuit designs!

1. Renesas Technology Corp. puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage. Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of non-flammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

1. These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corp. product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corp. or a third party.
2. Renesas Technology Corp. assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
3. All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corp. without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corp. or an authorized Renesas Technology Corp. product distributor for the latest product information before purchasing a product listed herein.
   The information described here may contain technical inaccuracies or typographical errors. Renesas Technology Corp. assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors.
   Please also pay attention to information published by Renesas Technology Corp. by various means, including the Renesas Technology Corp. Semiconductor home page (http://www.renesas.com).
4. When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corp. assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
5. Renesas Technology Corp. semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corp. or an authorized Renesas Technology Corp. product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
6. The prior written approval of Renesas Technology Corp. is necessary to reprint or reproduce in whole or in part these materials.
7. If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.
   Any diversion or reexport contrary to the export control laws and regulations of Japan and/ or the country of destination is prohibited.
8. Please contact Renesas Technology Corp. for further details on these materials or the products contained therein.

# Contents

Section 1 Specifications of the E8 Emulator ..... 1
Section 2 Connecting the Emulator with the User System ..... 3
Section 3 Pin Assignments of the E8 Connector ..... 5
Section 4 Example of E8 Connection ..... 7
Section 5 Differences between the MCUs and the Emulator ..... 11
Section 6 Applicable Tool Chain and Partner Tools ..... 19

# Section 1 Specifications of the E8 Emulator

Table 1.1 shows the specifications of the M16C/62P Group E8 Emulator.

Table 1.1 Specifications of the M16C/62P Group E8 Emulator

| | |
|---|---|
| Target MCU | M16C/ Family M16C/60 Series<br>M16C/62P Group |
| Usable MCU mode | Single-chip mode |
| Break function | - Address-match break, 8 points<br>- PC break (up to 255 points)<br>- Forcible break |
| Trace function | Not available |
| Flash memory programming function | Available |
| User interface | Clock-synchronized serial (communicating via P64/P65/P66/P67) |
| Program for the E8 Emulator | ROM size: 2 KB, RAM size: 128 bytes |
| Emulator power supply | Unnecessary (USB bus powered, power supplied from the PC) |
| Interface with host machine | USB (USB 1.1, full speed)<br>* Also connectable to host computers that support USB 2.0 |
| Power supply function | Can supply 3.3 V or 5.0 V to the target board (300 mA, max) |
| Power voltage | 3.0--3.6 V, 4.5--5.5V<br>* Available only when VCC1=VCC2 |

# Section 2 Connecting the Emulator with the User System

Before connecting an E8 emulator (hereafter referred to as emulator) with the user system, a connector must be installed in the user system so that a user system interface cable can be connected. When designing the user system, refer to Figure 3.1, Pin Assignments of the E8 Connector, and Figure 4.1, Example of E8 Connection, shown in this manual.
Before designing the user system, be sure to read the E8 emulator user's manual and the hardware manual for related MCUs.

Table 2.1 shows the recommended connector for the emulator.

Table 2.1 Recommended Connector

| Type Number | Manufacturer | Specifications |
|---|---|---|
| 2514-6002 | 3M Limited | 14-pin straight type |

Connect pins 2, 6, 10, 12, and 14 of the user system connector to GND firmly on the PCB. These pins are used as electrical GND and to monitor the connection of the user system connector. Note the pin assignments of the user system connector.

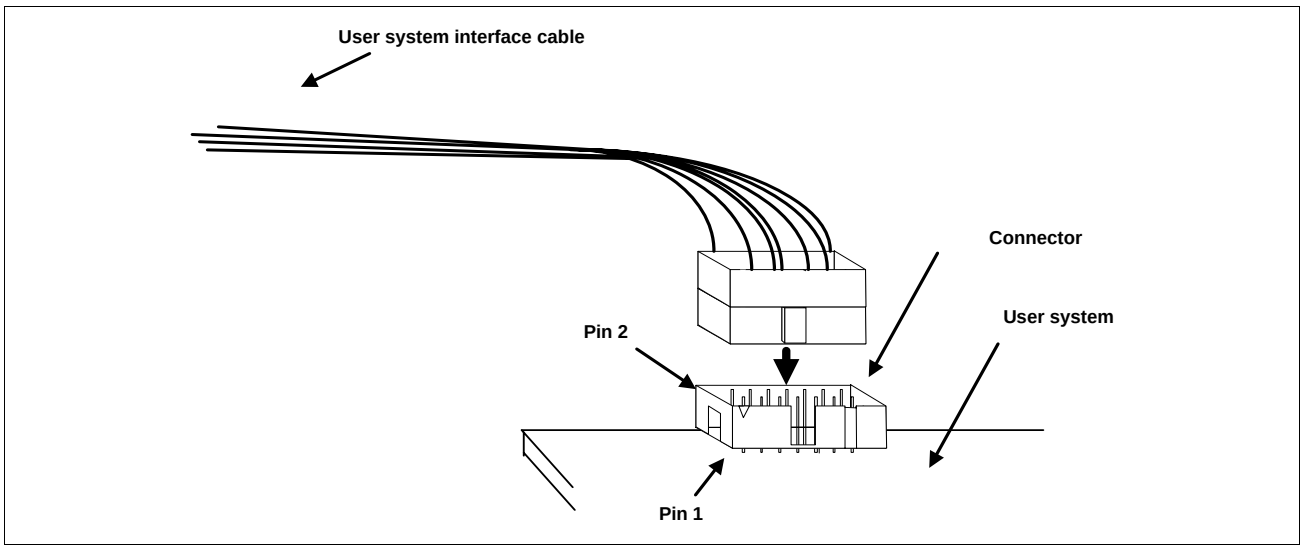
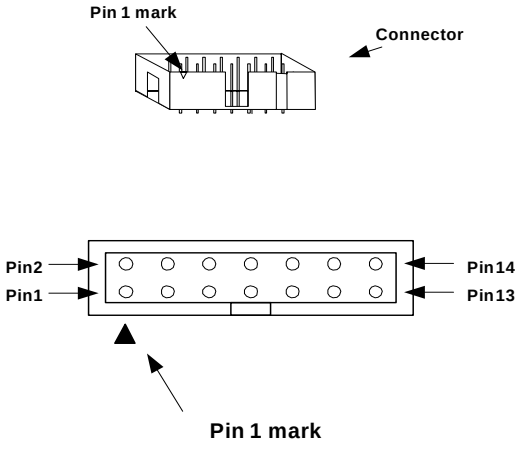



Figure 2.1 Connecting the User System Interface Cable to the User System

Notes: 1. Do not place any components within 3 mm of the connector.
2. When the emulator is used in the writer mode, connect the emulator similarly to the user system.

# Section 3 Pin Assignments of the E8 Connector

Figure 3.1 shows the pin assignments of the connector.




| Pin NO. | **M16C/62P** MCU signals |
|---|---|
| 1 | P65(SCLK) |
| 2 | Vss |
| 3 | CNVss |
| 4 | P55(EPM) |
| 5 | P67(TxD) |
| 6 | Vss |
| 7 | P50(CE) |
| 8 | Vcc |
| 9 | P64(BUSY) |
| 10 | Vss |
| 11 | P66(RxD) |
| 12 | Vss |
| 13 | $\overline{\text{RESET}}$ |
| 14 | Vss |

Figure 3.1 Pin Assignments of the E8 Connector

# Section 4 Example of E8 Connection

Figure 4.1 shows the connecting example.

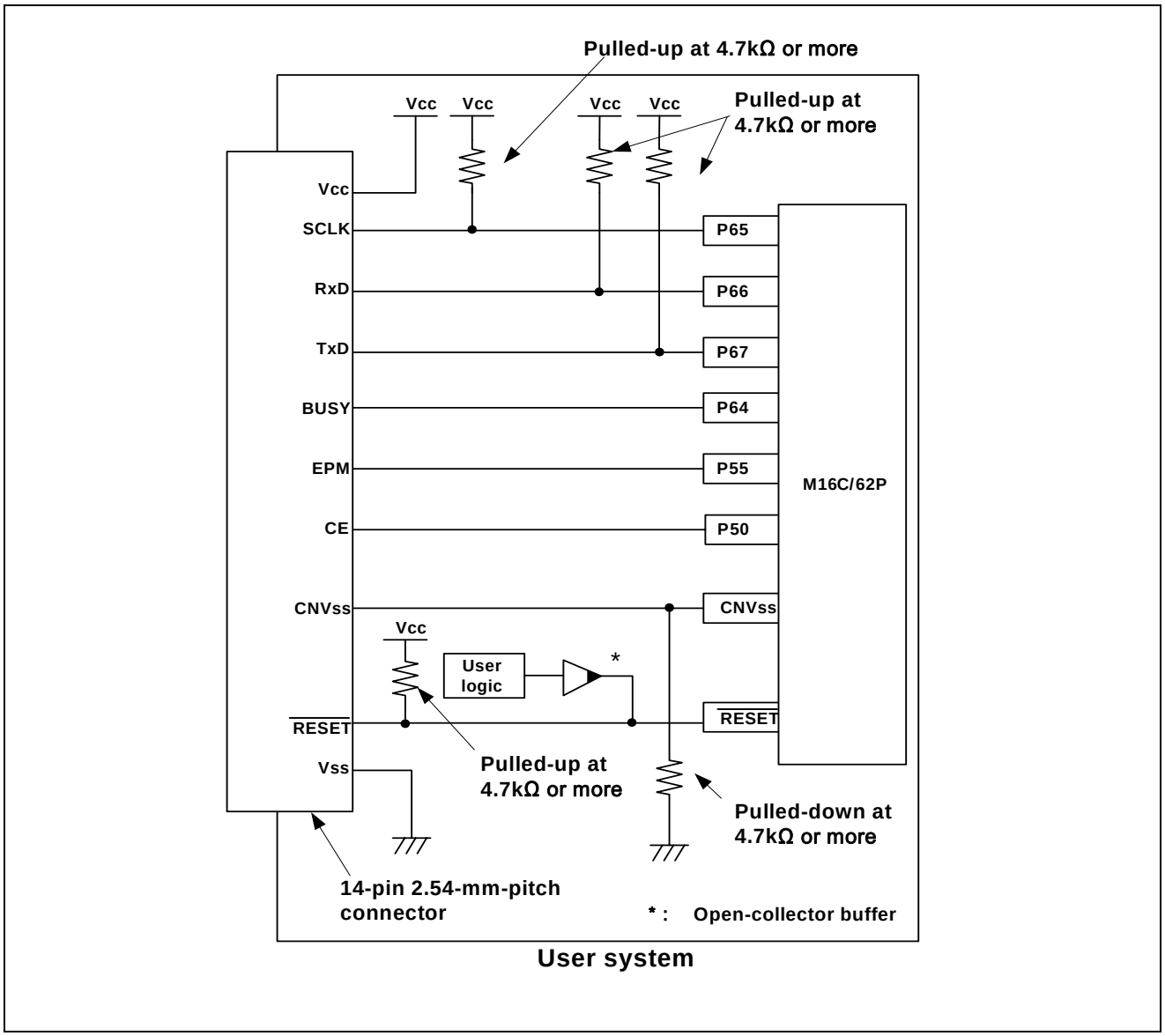



Figure 4.1 Example of E8 Connection

In the ‘Writing Flash memory’ mode, where the user program is simply written to the flash memory, the specification of connection between the E8 and the MCU is the same as that shown in Figure 4.1.

Notes: 1. P64, P65, P66 and P67 pins are used by the E8 emulator. Connect the E8 emulator to the MCU pins. For MCU pins P65, P66 and P67, pull up and connect to the emulator.

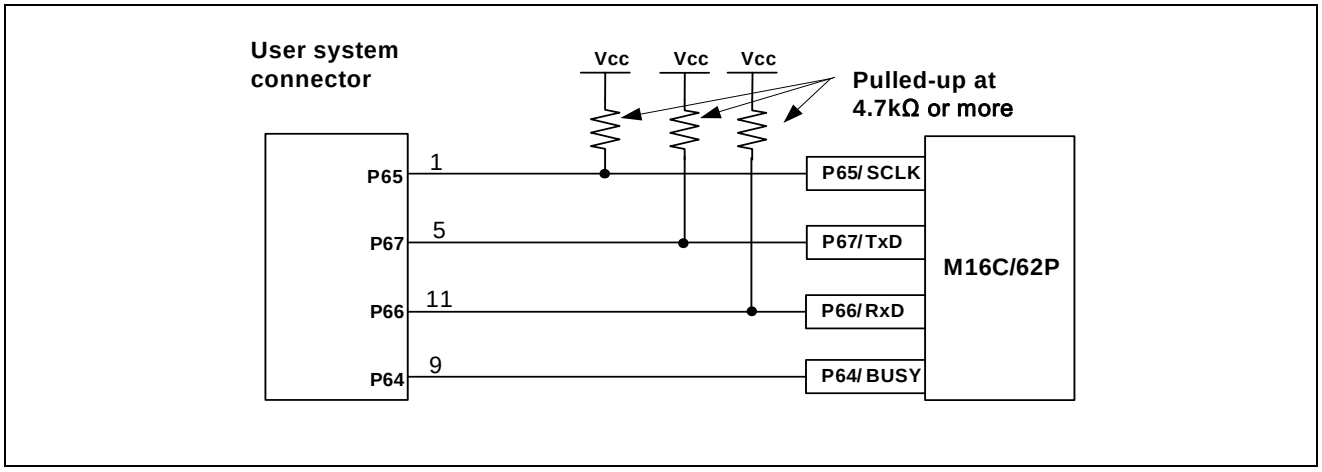



Figure 4.2 Connection of E8 Emulator and MCU

2. The E8 emulator uses the P50 and P55 pins for the MCU control. Connect the E8 emulator to the MCU pins.

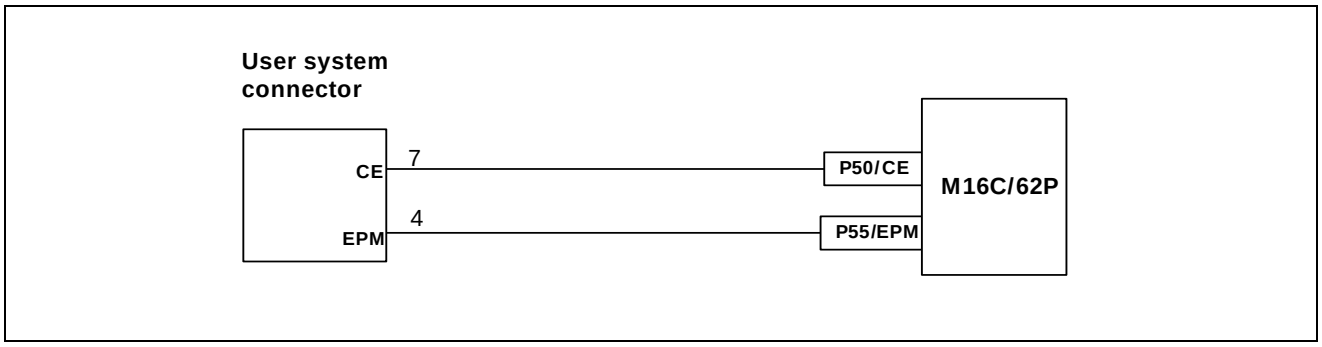



Figure 4.3 Connection of E8 Emulator and P50 and P55 Pins

3. The E8 emulator uses the CNVss pin for the MCU control. Connect the E8 emulator to the MCU pins through pull-down.

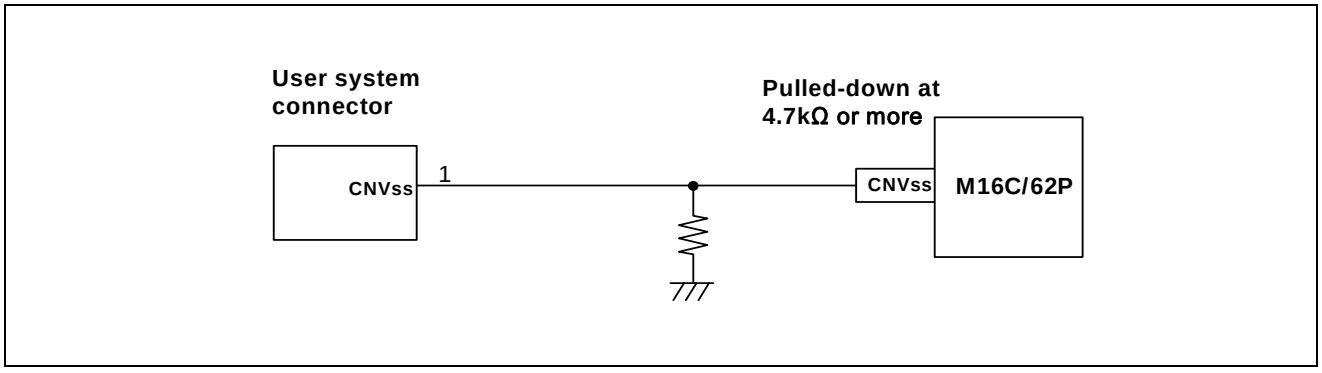



Figure 4.4 Connection of E8 Emulator and CNVss Pin

4. The $\overline{\text{RESET}}$ pin is used by the E8 emulator. Create the following circuit by connecting the open-collector output buffer so that reset input can be accepted from the E8 emulator.

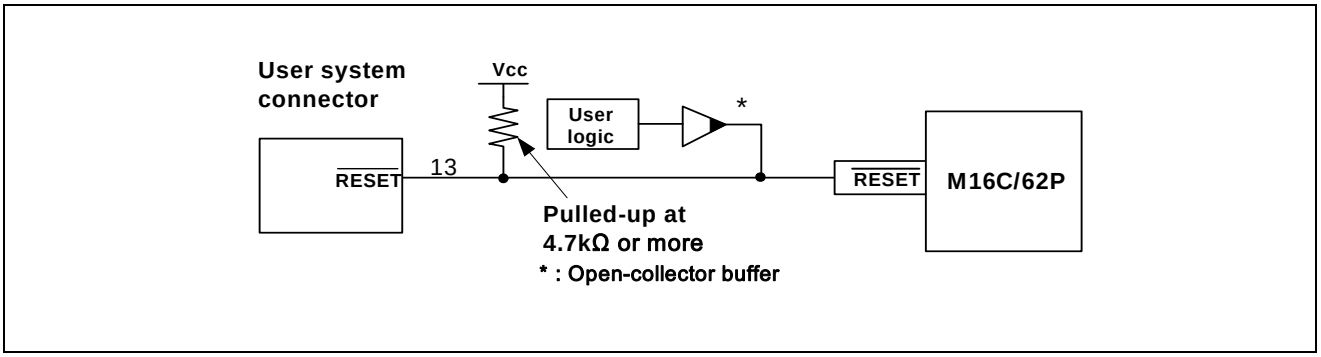



Figure 4.5 Example of a Reset Circuit

5. Connect Vss and Vcc with the Vss and Vcc of the MCU, respectively.
6. Connect nothing with N.C.
7. The amount of voltage permitted to input to Vcc must be within the guaranteed range of the microcomputer.

8. Figure 4.6 shows the interface circuit in the E8 emulator. Use this figure as a reference when determining the pull-up resistance value.

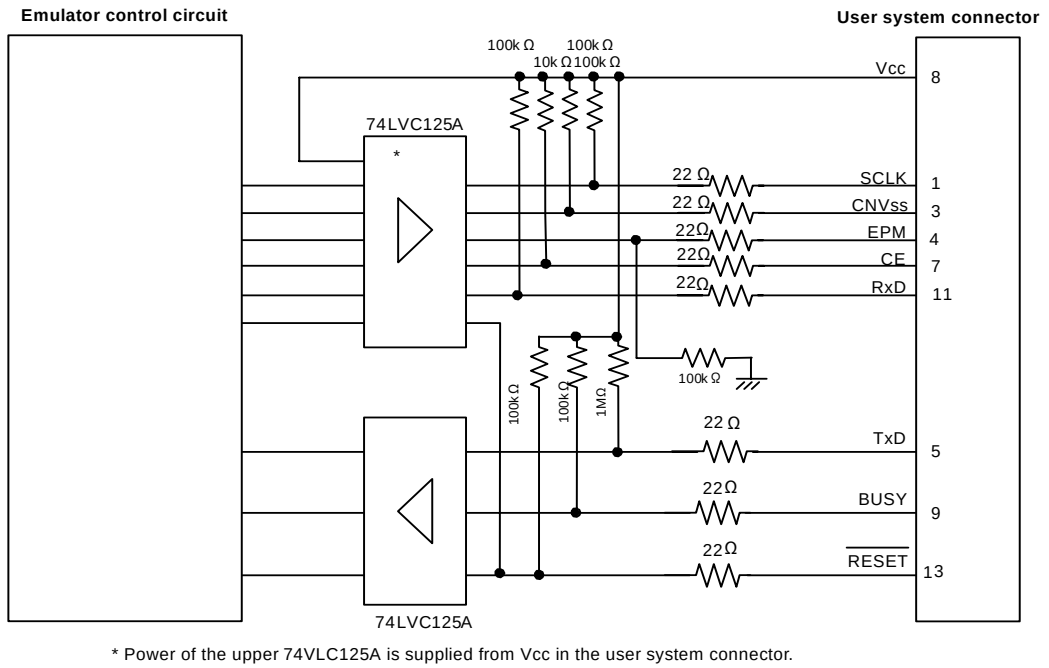



Figure 4.6 Interface Circuit in the Emulator (Reference)

# Section 5 Differences between the MCUs and the Emulator

1. Program area for the E8 emulator

   Table 5.1 lists the program area for the E8 emulator.

   Do not change this area, otherwise the E8 emulator will not operate normally. In this case, restart the High-performance Embedded Workshop in 'Download emulator firmware' mode.

Table 5.1 Program Area for the E8 Emulator

| Group | Type Number | ROM Size | | RAM Size | Program Area for E8 Emulator | | |
|---|---|---|---|---|---|---|---|
| | | Programming Area | Data Area | | Vector Area | ROM Area | RAM Area |
| M16C/62P | M30620FCP | 128KB | 4KB | 10KB | FFFE4h--FFFE7h, FFFE8h--FFFEBh, FFFECh--FFFEFh, FFFF4h--FFFF7h, FFFFCh--FFFFFh | 2 KB of the programming area [*1] | 128 bytes [*1] [*2] |
| | M30621FCP | 128KB | | 10KB | | | |
| | M30622F8P | 64KB | | 4KB | | | |
| | M30623F8P | 64KB | | 4KB | | | |
| | M30624FGP | 256KB | | 20KB | | | |
| | M30625FGP | 256KB | | 20KB | | | |
| | M30626FHP | 384KB | | 31KB | | | |
| | M30626FJP | 512KB | | 31KB | | | |
| | M30627FHP | 384KB | | 31KB | | | |

*1: When the High-performance Embedded Workshop is used in 'Download emulator firmware' mode, the dialog box shown in Figure 5.1 is displayed. Specify an area which is not used in the user system.

When the High-performance Embedded workshop is started with 'Does not download emulator firmware' mode, program area for the E8 emulator cannot be changed because the previous setting remains effective. When you change the program area for the E8 emulator, restart the High-performance Embedded Workshop in 'Download emulator firmware' mode.

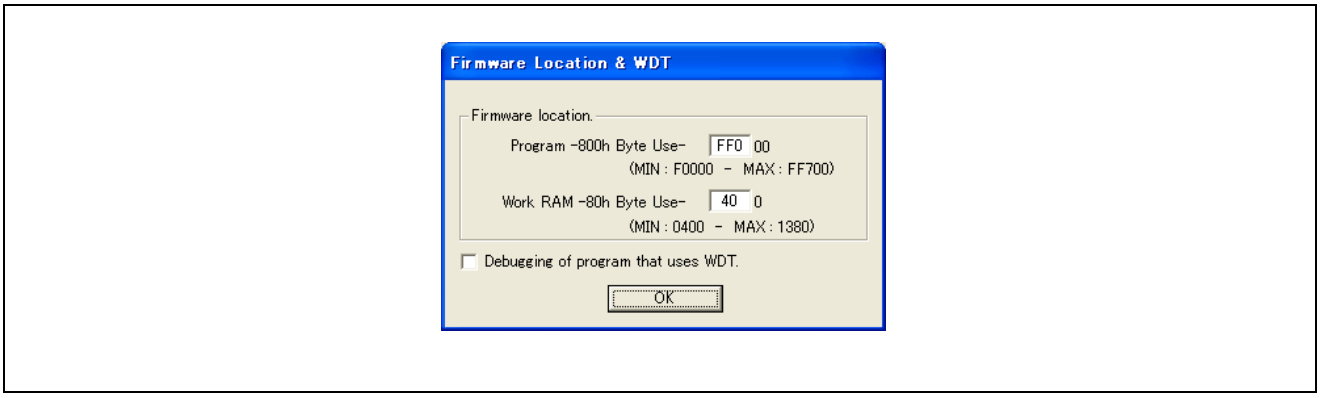

Firmware Location & WDT
Firmware location.
Program -800h Byte Use- FF0 00
(MIN : F0000 - MAX : FF700)
Work RAM -80h Byte Use- 40 0
(MIN : 0400 - MAX : 1380)
Debugging of program that uses WDT.
OK

**Figure 5.1 [Firmware Location & WDT] Dialog Box**

*2: RAM area used by the M16C E8 emulator debugger V.1.01.00 or later is 128 bytes. The former versions of emulator debuggers use 256 bytes.

2. Debugging of the watchdog timer
   When debugging the user program using the watchdog timer, select the [Debugging of program that uses WDT] check box in the [Firmware Location & WDT] dialog box. By selecting this box, the watchdog timer is being refreshed during the operation of the program for the E8 emulator. Note that if a memory is accessed by the memory reference or modification, the watchdog timer will be refreshed by the program for the E8 emulator.

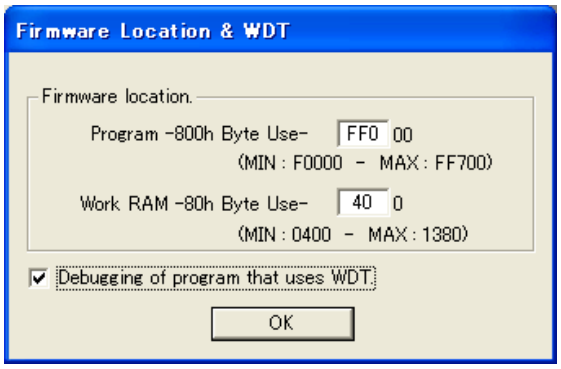

Firmware Location & WDT
Firmware location.
Program -800h Byte Use-
FF0 00
(MIN : F0000 - MAX : FF700)
Work RAM -80h Byte Use-
40 0
(MIN : 0400 - MAX : 1380)
Debugging of program that uses WDT
OK

Figure 5.2 [Firmware Location & WDT] Dialog Box

Note: When the High-performance Embedded workshop is started in 'Does not download emulator firmware' mode, the setting of the check box shown above cannot be changed because the previous setting remains effective. When you change the setting of the check box of the [Debugging of program that uses WDT], restart the High-performance Embedded Workshop in 'Download emulator firmware' mode.

3. ID code of flash memory

When the 7 bytes ID code (Table 5.2) written to the flash memory is other than FFh, FFh, FFh, FFh, FFh, FFh, FFh, input the ID code into the dialog box shown in Figure 5.3 which is displayed when starting up the High-performance Embedded Workshop.

When debugging in 'Download emulator firmware' mode or 'Does not download emulator firmware' mode, FFh, FFh, FFh, FFh, FFh, FFh, FFh is written into the ID code area regardless of the contents of the user program. In 'Writing flash memory' mode, the contents of the user program are input into the ID code area.

Table 5.2 ID Code Storage Area of M16C/62P

| Address | Description |
|---|---|
| FFFDFh | First byte of ID code |
| FFFE3h | Second byte of ID code |
| FFFEBh | Third byte of ID code |
| FFFEFh | Fourth byte of ID code |
| FFFF3h | Fifth byte of ID code |
| FFFF7h | Sixth byte of ID code |
| FFFFBh | Seventh byte of ID code |

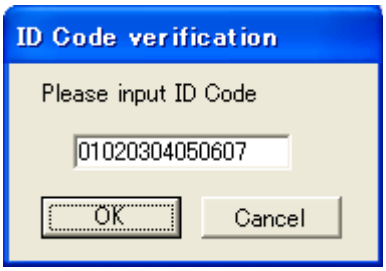



Figure 5.3 [ID Code verification] Dialog Box

[Note on Writing Flash memory mode]

When the ID code is specified by the -ID option of the lmc30, download the MOT file or HEX file. When the X30 file is downloaded, the ID code is not effective. When downloading the X30 file, specify the ID code using an assembler directive command such as ".BYTE". The file to which the ID code specified by the assembler directive command ".ID" is output varies depending on the version of the assembler. For details, refer to the user's manual of the assembler.

4. When the emulator system is initiated, it initializes the general registers and part of the control registers as shown in Table 5.3.

Table 5.3 Register Initial Values at Emulator Power-On

| Status | Register | Initial Value |
|---|---|---|
| Emulator Power-On | PC | Reset vector value in the vector address table |
| | R0 to R3 (bank 0, 1) | 0000h |
| | A0, A1 (bank 0, 1) | 0000h |
| | FB (bank 0, 1) | 0000h |
| | INTB | 0000h |
| | USP | 0000h |
| | ISP | Work RAM Address for the E8 emulator + 80h * |
| | SB | 0000h |
| | FLG | 0000h |

Note: The Work RAM address for the E8 emulator is specified in [Firmware Location & WDT] dialog box when the High-performance Embedded Workshop starts up in “Download emulator firmware” mode.

5. Operation clock while the user program remains idle

   While the user program remains idle, the E8 emulator program changes the main clock divide-by-N value as it runs.

6. Reset

   The reset vector is used by the E8 emulator program. If the MCU is reset while executing the user program, control is transferred to the E8 emulator program and the user program is made to stop.

7. Memory access during emulation execution

   When referring or modifying the memory contents, the user program is temporarily halted. For this reason, realtime emulation cannot be performed.

8. The emulator controls the MCUs by using the P50, P55, P64, P65, P66, P67, $\overline{\text{RESET}}$ and CNVss pins.

9. The power consumed by the MCU increases by several mA or over 10 mA. This is because the user power supply drives one 74LVC125A to make the communication signal level match the user system power-supply voltage.

10. The emulator uses up to 14-byte stack pointer when a user program breaks. Accordingly, reserve the 14-byte addresses for the stack area.

11. When debugging, the flash memory is frequently re-written by the E8 emulator. Therefore, do not use an MCU that has been used for debugging.

    Also, as the program for the E8 emulator is written into the MCU while debugging, do not save the contents of the MCU’s flash memory that have been used for debugging or use them as the ROM data for products.

12. SFR used by the program for the E8 emulator

As the SFR listed in Table 5.4 is used by the program for the E8 emulator, do not change a value. Otherwise, the E8 emulator cannot be controlled. Also, they are not initialized by selecting [Debug] -> [Reset CPU] or with the RESET command. If their contents are referred to, a value that has been set in the program for the E8 emulator will be read.

Table 5.4 SFR Used by Program for E8 Emulator

| Address | Register | Symbol | Bit | Notes on using the E8 emulator |
|---|---|---|---|---|
| 03A8h | UART1 transmit/receive mode register | U1MR | All bits | [*1] |
| 03AAh, 03ABh | UART1 transmit buffer register | U1TB | All bits | [*1] |
| 03ACh | UART1 transmit/receive control register 0 | U1C0 | All bits | [*1] |
| 03ADh | UART1 transmit/receive control register 1 | U1C1 | All bits | [*1] |
| 03AEh, 03AFh | UART1 receive buffer register | U1RB | All bits | [*1] |
| 03B0h | UART transmit/receive control register 2 | UCON | Bits 1, 3, 4, 5 and 6 | [*2] |
| 03ECh | Port P6 register | P6 | Bits 4, 5, 6 and 7 | [*2] |
| 03EEh | Port P6 direction register | PD6 | Bits 4, 5, 6 and 7 | [*2] |

*1 Do not change the value of the register.

*2 Do not change the value of the bits listed above. When operating this register, change it by a bit operating instruction, etc.

13. Interrupts used by the E8 emulator program

The BRK instruction interrupt, address match interrupt, single-step interrupt, and DBC interrupt are used by the E8 emulator program. Therefore, make sure the user program does not use these interrupts.

14. NMI interrupt

If NMI interrupts are to be used, be sure to take the necessary measures before executing the user program by, for example, disabling automatic updates of the watch window and freezing the display of the memory window in order to ensure that no memory accesses will occur during user program execution.

If an NMI interrupt occurs while the user program remains idle or when memory contents are referenced or changed during user program execution, device operation becomes uncontrollable by the E8 emulator.

15. Reserved area

The addresses not specified in the Hardware Manual for M16C/62P Group is reserved area. Do not change the contents. Otherwise, the E8 emulator cannot be controlled.

16. Debugging in the stop mode or wait mode

When using the stop mode or wait mode on a user program, firstly disable the automatic update in the watch window or fix the display in the memory window so that the memory access will not occur during execution. In addition, do not operate the window until the program stops at the breakpoint by setting the breakpoint at the processing unit where the stop mode or wait mode is cancelled.

17. Peripheral I/Os during a break

During a break, although interrupts are not accepted, peripheral I/Os continue to be operated. For example, a timer interrupt is not accepted although counting a timer is continued when a user program is stopped by a break after operating a timer.

18. Exceptional step operation

a) Software-interrupt instruction

STEP operation cannot be performed by continuously executing the internal processing of instructions (undefined, overflow, BRK, and INT) which generates a software interrupt.

<Example> INT instruction

```
            NOP
            NOP
            INT#3
            NOP          ↘ Passes through if the STEP operation is carried out.
            JMP MAIN
INT_3:
            NOP  ←  The address at which the program should be stopped.
            NOP
            NOP
            REIT
```

b) INT instruction

Debugging of the program using the INT instruction should be used with the GO command by setting a software break for the internal processing of the INT instruction.

<Example>

```
            NOP
            INT  #3
            NOP                    ↓ Execution with the GO command
            JMP  MAIN
INT_3:
NOP  Break  ←
NOP
REIT
```

19. “Run to cursor” function

The "Run to cursor" function is realized by using an address match break. Therefore, when you execute the "Run to cursor" command, all the address match breaks you set become invalid, while all the PC breaks remain valid.

20. Note on PC break point

When downloading a user program after changing it, the address setting of a PC break may not be corrected normally depending on the changes. After downloading a user program, please check the setting of a PC break by event point window and reset it.

21. Note on debugging in CPU rewrite mode

When debugging in CPU rewrite mode, do not rewrite the CPU’s block 0 area (addresses FF000h – FFFFFh) and block containing the program for the E8 emulator. If these areas are rewritten, the E8 emulator will run out of control.

Do not halt the user program after setting the CPU rewrite mode until releasing it. If you do so, the E8 emulator may run out of control. Cancel the automatic renewal in the watch window in advance and select fixing display in the memory window to prevent a memory access from occurring while executing the user program.

To check the data after executing the CPU rewrite mode, halt the program after releasing the CPU rewrite mode and see the memory window etc.

22. Note on lock bits of flash memory

When starting up the High-performance Embedded Workshop in ‘Download emulator firmware’ mode or ‘Writing Flash memory’ mode, lock bits in all the blocks of the flash memory will be unlocked.

Note that the lock bits of the downloaded blocks will be unlocked after downloading the user program.

# Section 6 Applicable Tool Chain and Partner Tools

With the M16C/62P Group E8 emulator, you can debug a module created by the inhouse tool chain and third-party products listed in Table 6.1 below.

Table 6.1 Applicable Tool Chain and Partner Tools

| | |
|---|---|
| Tool chain | M3T-NC30WA V.5.20 Release 1 or later |
| Partner tools | TASKING M16C C/C++/EC++ Compiler V.2.3r1 or later<br>IAR EWM16C V.2.12 or later |

[Precautions on debugging the load modules created in ELF/DWARF2 format]

If the load module was created in ELF/DWARF2 format using TASKING M16C C/C++/EC++ compiler V3.0r1, the precaution described below must be observed when displaying member variables of the base class in the watch window.

<Precaution>

If any class object that has a base class is defined, the following problems may occur:

Case 1: Member variables of the base class cannot directly be referenced from the class object (*1).

Case 2: If the PC value resides in any member function of a derived class, member variables of the base class cannot directly be referenced (*4).

<Solution>

If member variables of the base class need to be referenced in the watch window, follow either method described below.

Case 1: Use indirect references from the class object to refer to member variables of the base class (*2) (*3).

Case 2: Use indirect references from “this” pointer to refer to member variables of the base class (*5) (*6).

<Example program statement>

```
////////////////////////////////////////////////////////
*.h
    class BaseClass
    {
    public:
        int m_iBase;
    public:
        BaseClass() {
            m_iBase = 0;
        }
        void BaseFunc(void);
    };

    class DerivedClass : public BaseClass
    {
    public:
        int m_iDerive;
    public:
        DerivedClass() {
            m_iDerive   = 0;
        }
        void DerivedFunc(void);
```

```
};
```

*.cpp

```
main()
{
    class DerivedClass ClassObj;
    ClassObj.DerivedFunc();
    return;
}

void BaseClass::BaseFunc(void)
{
    m_iBase = 0x1234;
}

void DerivedClass::DerivedFunc(void)
{
    BaseFunc();
    m_iDerive   = 0x1234;
}
```

////////////////////////////////////////////////////////

<Example for registering in the watch window>

////////////////////////////////////////////////////////

Case 1: If the PC value resides in the main() function

(1)"ClassObj.m_iBase" : Cannot be referenced (*1)

(2)"ClassObj.__b_BaseClass.m_iBase" : Can be referenced (*2)

(3)"ClassObj"

-"__b_BaseClass"

-"m_iBase" : Can be referenced (*3)

-"m_iDerive"

-: Expansion symbol

Case 2: If the PC value resides in the DerivedClass::DerivedFunc() function

(1)"m_iBase" : Cannot be referenced (*4)

(2)"this->__b_BaseClass.m_iBase" : Can be referenced (*5)

(3)"__b_BaseClass.m_iBase" : Can be referenced (*5)

(4)"this"

-"*"

-"__b_BaseClass"

-"m_iBase" : Can be referenced (*6)

-"m_iDerive"

(5)"__b_BaseClass"

-"m_iBase" : Can be referenced (*6)

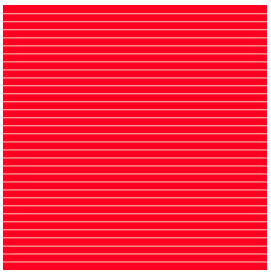
////////////////////////////////////////////////////////

E8 Emulator
Additional Document for User's Manual
Notes on Connecting the M16C/62P


| | | |
|---|---|---|
| Publication Date: | Apr. 1, 2005<br>Jun. 23, 2005 | Rev.1.00<br>Rev.2.00 |
| Published by: | Sales Strategic Planning Div.<br>Renesas Technology Corp. | |
| Edited by: | Microcomputer Tool Development Department<br>Renesas Solutions Corp. | |


© 2005. Renesas Technology Corp. and Renesas Solutions Corp., All rights reserved. Printed in Japan.

# E8 Emulator
# Additional Document for User's Manual



Renesas Technology Corp.
2-6-2, Ote-machi, Chiyoda-ku, Tokyo, 100-0004, Japan